DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW

Final Project

Problem Description

I wanted to experiment with computer vision in Python and this project gave me a good way to do so. I chose to work on classification with computer vision in a simple problem domain: training a pair of models to recognize hand gestures in the game of rock paper scissors and predict what move the player will choose next based on their prior moves. The resulting system could play rock paper scissors with the player and display the results of the game.

Core Problem

Playing rock paper scissors breaks down to a few fundamental components:

1. Getting an image of the player's hands
2. Classifying that image as rock, paper, or scissors
3. Predicting what the player is about to do based on historical patterns
4. Choosing a move that will beat what the AI expects the player to choose
5. Evaluating the winner based on the chosen move and the player's observed move
6. Recording player move choices to help train subsequent models

Additionally, all of this had to work in a single notebook without any data sources being embedded in the notebook.

Machine Learning Tasks

This problem required a pair of multi-class classification models to accomplish its tasks.

The first model is an image classification model, trained on a dataset of images of hand gestures for rock paper scissors. This model is used to classify images from the webcam or from disk to recognize the move the player selected.

The second model is a classical multi-class classification model predicting a target label of rock, paper, or scissors representing the player's next move based on their last two moves and who won the last two games. This model was specifically trained on dataset involving my own historical moves.

Model Features and Target Variables / Labels

The image classification model's features are a one-dimensional array of pixel data representing the image the model was trained on. These pixels were reduced to grayscale and then normalized within the context of the entire image's color spectrum. Additionally, images were resized down to a consistent size so that this array would be the same size for all images.

The label of the image classification model was rock, paper, or scissors and was based on the directory the image was located within the training data.

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW
The move history data is in the following format:

t_minus_1_player_move	t_minus_1_winner	t_minus_2_player_move	t_minus_2_winner	next_move
rock	tie			scissors
rock	tie			rock
rock	tie			rock
rock	player	rock	tie	scissors
scissors	player	rock	player	paper
paper	player	scissors	player	paper
paper	ai	paper	player	rock
rock	ai	paper	ai	rock
rock	ai	rock	ai	paper
paper	ai	rock	ai	paper
paper	player	paper	ai	scissors
scissors	tie	paper	player	paper
paper	ai	scissors	tie	scissors
				scissors
				scissors
scissors	ai			rock
rock	tie	scissors	ai	scissors

Here the features are the player's last move, who won that game, the player's move the game before that, and who won that game. The label being predicted is the move the player is about to select.

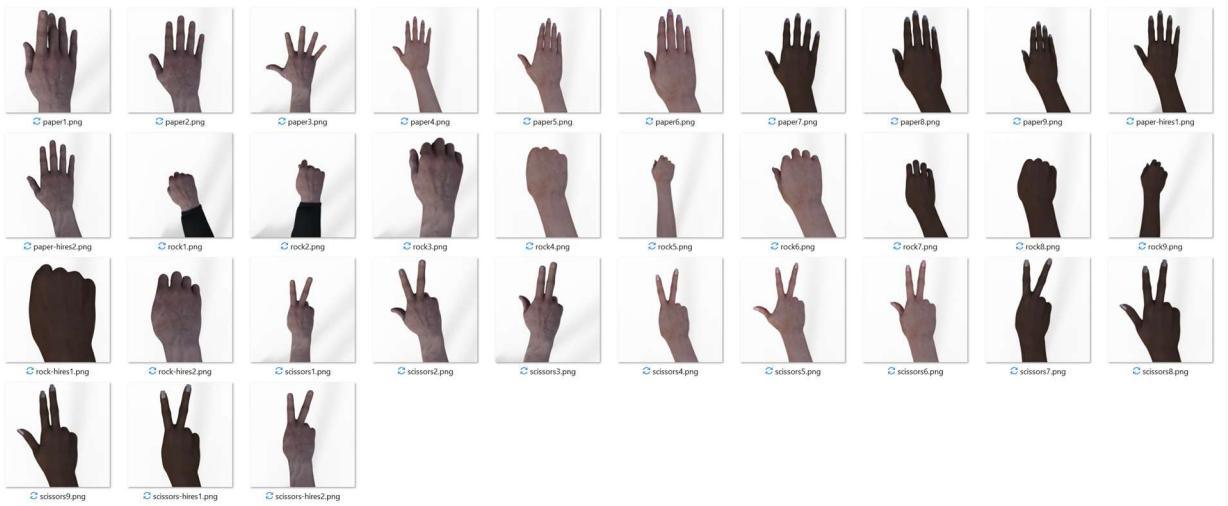
This data is added to as the game is played, so whenever the player selects a move, that move is recorded as a new label, with features recorded for the game before and the game before that. When no game was played previously (the first two plays of a session, for example), empty values are used to represent those games. This helps predict the player's initial choices the first game when no other game choices are available.

Data Exploration & Pre-Processing

Dataset

The assignment dictated a third-party data source, so I went with Sanikamal's Rock Paper Scissors Dataset at [Rock Paper Scissors Dataset | Kaggle] with a sample set of images pictured below:

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW



Because the image data was very large, the Jupyter Notebook will download and unzip a zipped version of the dataset from Kaggle when the application runs if that data is not already present locally. Because Kaggle requires authentication to download files, I hosted the zip file on my Azure account in Blob storage instead, but it is the same core data source as the version on Kaggle.

Data for game moves could not be found in the format I desired so it is created and recorded as the application is used. See the prior section for information about how this data is recorded. I also have the application download a sample data file from blob storage if this historical data is not present.

Data Exploration

Data exploration was less relevant for me for this project because the initial series of images is not one that yields itself well to charting. However, I looked through the images and observed a variety of skin tones and styles of representing rock, paper, and scissors. I also observed a consistent white background and hands coming from the bottom of the frame. This helped me design my own setup for capturing images as shown below:

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW

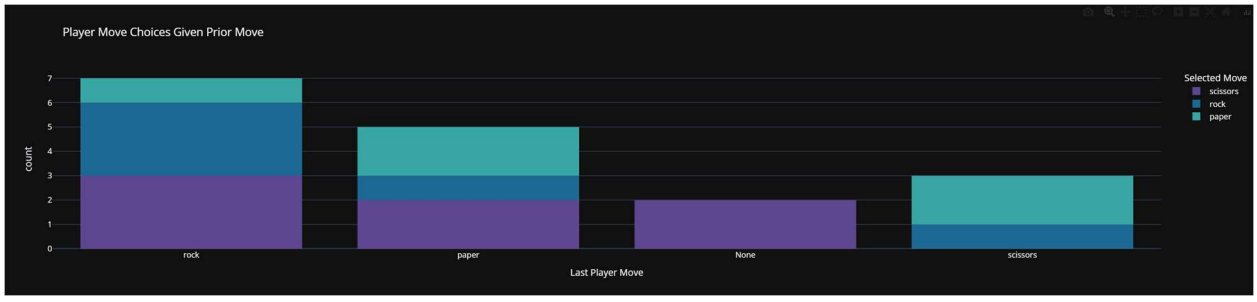
Here I use my recording lights and the cream wall as the best proxy available for a white background. I also use a webcam on a clamp to help reproduce the angle in the shots. This helps me produce images similar to the following:

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW

My second dataset was one I created as the application ran, so it did not require data visualization before the experiment ran, however I was able to perform some visualization experiments with the resulting data exposing some of my biases in move choices overall:

Here I seem to favor scissors overall, but not by too much.

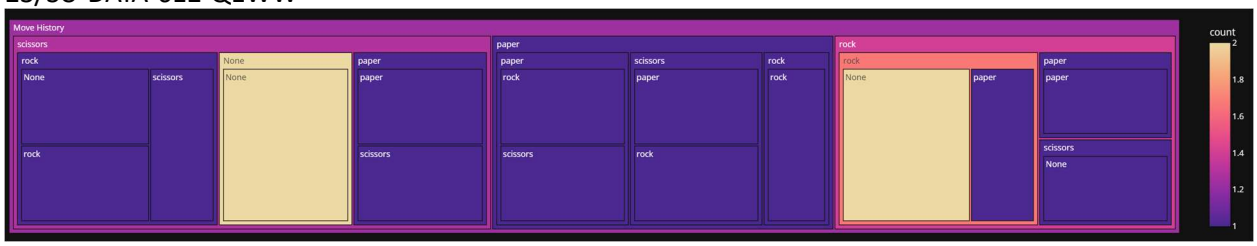
I also was able to spot some trends in move choices based on the prior move choices:



This insight shows me that I tend to open with scissors and tend to move from paper and rock to scissors, while also tending to stay more on rock than I do with other move choices.

This data could be charted further in hierarchical manner up to the last two choices:

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW



This isn't incredibly insightful though it does reveal my tendency to remain on rock perhaps more than any other choice.

Overall, the patterns in the data visualization do illustrate some trends that a classification model could identify, which does validate my approach.

Data Cleaning & Preparation

The primary dataset was an image dataset and so it didn't have the normal problems you'd find involving missing values, typos, and general data needing to be cleaned. However, the nature of the images required a lot of pre-processing of these images to reduce the noise and variety in the images and help the deep learning model focus on the core image features.

Ultimately, I chose a number of image processing steps that took a source image like this:

And reduced it down to something simpler like this:

The ordered steps I took to process the image follows:

1. Reduce the image to grayscale
2. Resize the image to 100 x 100 pixels
3. Normalize pixel brightness in the image so the darkest in each image should be 0 and the brightest has a level of 255 (maximum brightness in a color scale)
4. Apply fastN1MeansDenoising using OpenCV to remove any background variation from shadows or uneven lighting
5. Apply morphologyEx in OpenCV to help blend together shapes in the hand and reduce the odds of random features appearing in the resulting image on the back of the hand
6. Apply a median blur for the same reasoning
7. Apply Canny edge detection to remove all hand and background data and focus only on the outline of the hand
8. Flatten the image data down to a one-dimensional array

This processing function applied to any image used for training or testing the model as well as any image grabbed from the webcam later.

The dataset of game moves didn't require too much processing other than I needed to one-hot encode categorical values such as rock / paper / scissors before training the model and generating predictions. Notably this dataset was entirely categorical with data for moves selected and wins, losses, and ties.

I did need to do an additional bit of data processing to generate the treemap visuals. I needed to handle missing values as "None" representing no move selected / no prior game, and I needed to engineer a new column to count the number of rows identical to a given row in the dataset and then drop duplicated rows. This then let me build the treemap with reliable count values.

Modelling

Approach: Random Seed and Test / Train Split

I included a random seed in the model training process. As far as the test / train split, the images dataset was already partitioned into a test / train set of directories, so I elected to keep that partitioning instead of writing python code to manually join them all together and then split at random. However, the move prediction logic does use the 20% test data split technique.

Machine Learning Algorithms

In focusing on image classification, I tried a large number of algorithms inside of the SciKit-Learn library before settling on MLPClassifier as a means of using a neural network to attempt to make sense of the image data.

I was disappointed with the model's performance for some time, and so I wound up using a GridSearch to find an optimal set of hyperparameters for this model training process. The best performance I found wound up using logistic activation, hidden layers of 100, 50, and 25, and a SGD solver.

Note: grid search takes several hours with this dataset and I do not recommend running it yourself

My focus was primarily on the image classification model and once I was satisfied with its performance, I more or less decided to use the same algorithm on the move prediction side for convenience purposes,

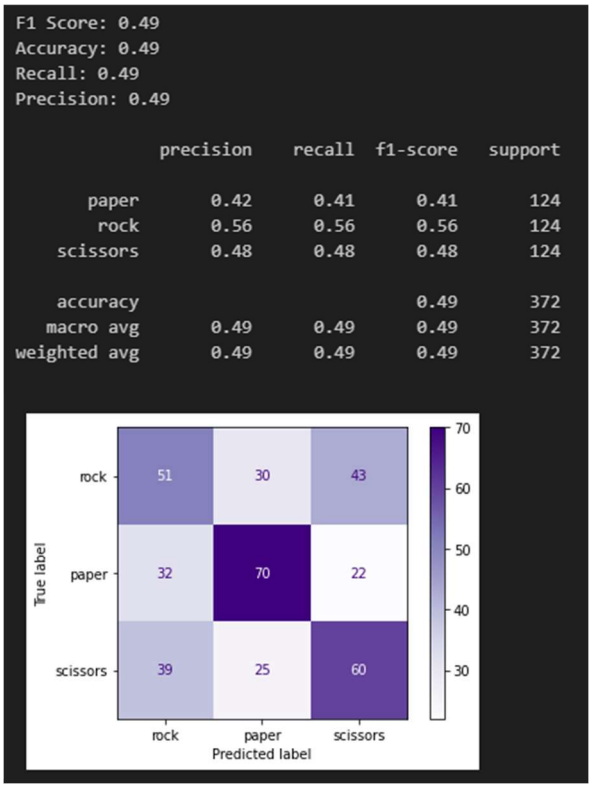
though I suspect that a far simpler LogisticRegression would have been almost as good, given the simplicity of the dataset.

Model Evaluation

In any classification approach, I tend to prefer the whole confusion matrix over any individual metric, though some models I favor precision while others I favor recall. If I had to pick a single important metric for this particular model, I'd probably pick the F1 Score as a good overall evaluation metric.

Model Performance

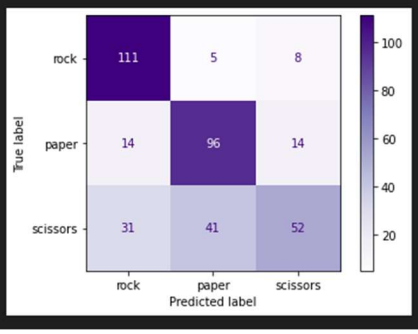
In general, the model performance for the image classification model was below my expectations, with a 0.49 F1 score for the final model I selected.



Other models had better metrics, but I was actually quite happy with how this particular model performed against my own hand gestures, leading me to select it over some of the early suggestions from grid search:

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW

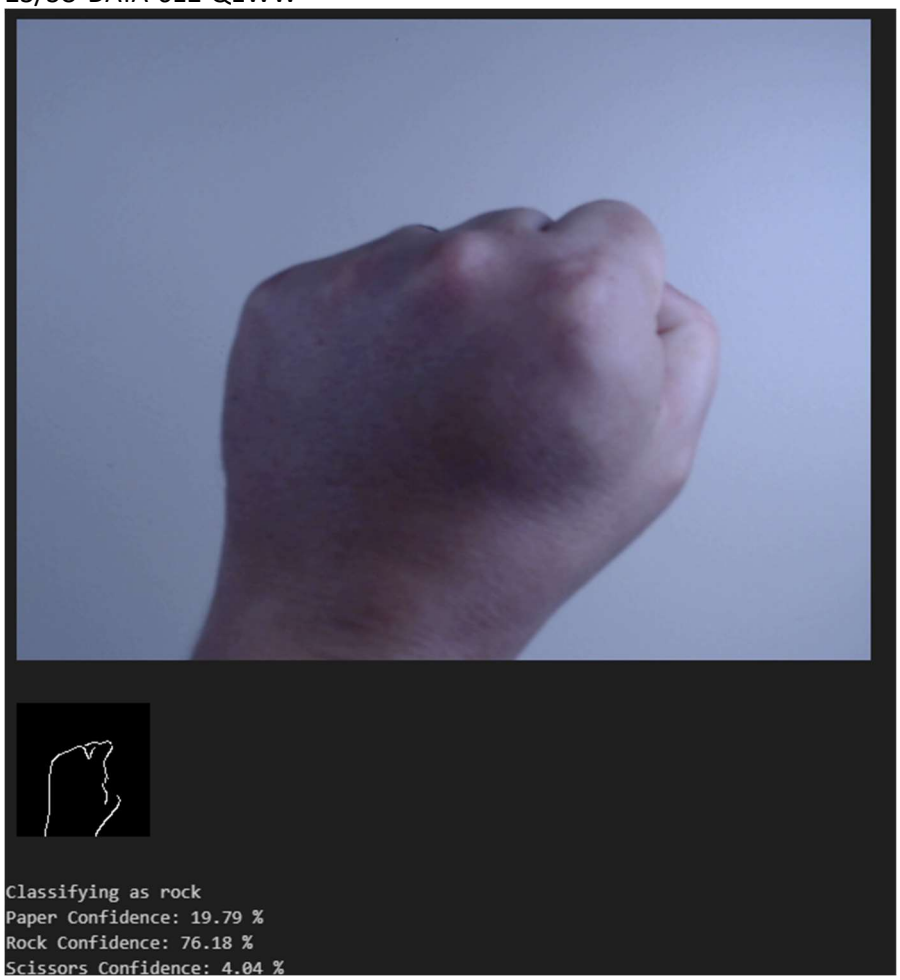
```
Finding best model with grid search. This is going to take a few hours...
Fitting 2 folds for each of 108 candidates, totalling 216 fits
Best parameters: {'activation': 'logistic', 'hidden_layer_sizes': (100, 50, 25), 'learning_rate': 'constant', 'solver': 'sgd'}
Accuracy: 0.70
F1 Score: 0.68
Accuracy: 0.70
Recall: 0.70
Precision: 0.70

              precision    recall  f1-score   support

       paper       0.71      0.90      0.79       124
        rock       0.68      0.77      0.72       124
    scissors       0.70      0.42      0.53       124

    accuracy                           0.70       372
   macro avg       0.70      0.70      0.68       372
weighted avg       0.70      0.70      0.68       372

c:\Users\Admin\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic Optimizer
  warnings.warn(
```



From this difference between the “optimal” performance metrics and the model I preferred, I can infer that my filming setup for hand gesture recognition or my style of indicating different hand signals is significantly different than the environment / gesture styles used by the people creating the dataset.

In particular, I noticed that my style of indicating rock is different than most of the training data, for example, but my model does well at interpreting it:

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW

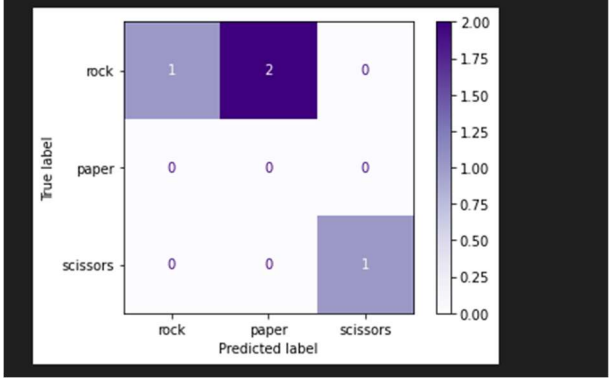


This finding does not make me feel self-conscious playing rock paper scissors *at all*!

As far as the metrics for the move prediction model goes, it suffered a bit from a small dataset size as indicated by the very minimal confusion matrix below:

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW

```
F1 Score: 0.62
Accuracy: 0.50
Recall: 0.50
Precision: 1.00

              precision    recall  f1-score   support

       paper       1.00      0.33      0.50         3
        rock       0.00      0.00      0.00         0
    scissors       1.00      1.00      1.00         1

    accuracy                           0.50         4
   macro avg       0.67      0.44      0.50         4
weighted avg       1.00      0.50      0.62         4
```



I think that if my filming setup was more convenient for game play that it would have been easier to record a larger variety of moves which would have resulted in a more compelling set of metrics.

That being said, my anecdotal evidence is that this model usually beats me, sometimes ties me, and only occasionally loses to me.

Since my core goal was to recognize my own hand gestures and predict against my own moves, I feel these models have met those goals for this particular usage scenario.

Project Results and Key Takeaways

As I mentioned a moment ago, I do view this project as a success, albeit a very limited one. I would have liked more universally compelling metrics on the image classification model, and I would have liked to not need a challenging and unergonomic filming setup to record my hand motions, but image classification turns out to be fairly difficult to get right.

Some of my key takeaways from this experiment include:

- Deep learning models can key in on the smallest things and so controlling the environment is paramount
- Your training data is important. In this case I was not part of any of the training data and my environment and hand gestures were both significantly different
- The volume of data matters, particularly if you haven't hit a minimum threshold of rows of data
- The steps you take in processing images and their order has a huge impact on the resulting features
- Reducing noise in images through blurring and normalization is very helpful

DATA-612 Final Project
Matt Eland (0771584)
23/SU-DATA-612-Q1WW

- Canny edge detection is fantastic, but only if you clean up images beforehand
- Canny edge detection takes a lot of time
- Grid search can be very helpful in optimizing hyperparameters
- I tend to think I don't play scissors very much and so I over-use it to compensate for this perceived deficiency

If I had to repeat this experiment, I would likely use a hand gesture landmarking library like the one at https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer/python. This processing step would identify landmarks in hands and these landmarks could be used for model training instead of the raw image data.

I also would likely modify my code to record my own hand gestures to the training and test image directories so that as I used it and retrained the model periodically the model would be more inclined to recognize my own unique ways of indicating these moves.

Overall, it was a good experience and helped me learn and apply myself while exploring an area of interest.